1 Activity goal

The goal of this laboratory activity is to introduce some improvements to the speed PI control system designed in the laboratory activity 2. The improvements consist in the implementation of an anti–windup scheme to reduce the large overshoot occurring in the step response when the controller output saturates, and a friction plus inertia feedforward compensator, which allows to enhance both the accuracy and speed of response of the conventional feedback controller.

2 Feedforward compensation

The benefits of using feedback in a control system are well known; they include the ability of reducing the control sensitivity to parameter variations and external disturbances, the possibility of stabilising unstable plants, and the capability of enhancing both the control precision and speed of response. Nevertheless, the performance of a conventional feedback (i.e. closed–loop) control system is often improved by resorting to feedforward (i.e. open–loop) control actions. Feedforward actions are typically introduced for:

1. pre–filtering the reference signal in order to match the overall control system transfer function (from the reference input to the controlled output) to that of a desired reference model.

With reference to Fig. 1a, it holds that

\[ Y(s) = T'(s) H_1(s) R(s) \quad \text{with} \quad T'(s) = \frac{Y(s)}{R'(s)} = \frac{C(s)P(s)}{1 + C(s)P(s)} \]  \hspace{1cm} (1)

Therefore, the feedforward compensator (reference pre–filter) \( H_1(s) \) can be used to introduce some adjustments to the overall closed–loop system dynamics (from the reference input \( r \) to the controlled output \( y \)). In particular, it can be used to match the closed–loop response to that of a reference model with transfer function \( T_m(s) \) (under certain assumptions for \( T_m(s) \)).

2. improving the control accuracy and speed of response.

With reference to Fig. 1b, it holds that

\[ Y(s) = \frac{C(s)P(s) + H_2(s)P(s)}{1 + C(s)P(s)} R(s) \]  \hspace{1cm} (2)
For perfect tracking, it is desired to have $y(t) = r(t)$ for all $t \in \mathbb{R}$ and any reference signal $r(t)$. This specification cannot be guaranteed with a simple feedback controller. Indeed, what can be usually achieved with a simple feedback control is the perfect asymptotic tracking of specific reference inputs, namely the capability of following certain signals with zero tracking error at steady state. This is done by exploiting the internal model principle, namely by embodying a model of the signal to track (or the disturbance to reject) in the controller dynamics. For example, an integral action is typically included in the controller to achieve perfect steady state regulation of constant set-points (i.e. step reference inputs), and perfect steady state rejection of constant load disturbances (i.e. step disturbances entering at the plant input).

Perfect tracking “in the general sense” can be ideally achieved by employing a feedforward action: in fact, from (2), it follows that the perfect tracking condition $Y(s) = R(s)$ can be satisfied by choosing

$$H_2(s) = \frac{1}{P(s)}$$

Obviously, this choice is not always practicable. In particular, it is requested that

- $P(s)$ does not contain zeros with positive real part (i.e. minimum-phase condition).
- $P(s)$ is a proper transfer function.
- the plant dynamics does not contain relevant input–output (I/O) time–delays.

However, some conditions can be relaxed, on condition of modifying the structure of the feedforward compensator. In fact

- strictly proper transfer functions can be tolerated, provided that the time derivatives of the reference signal (up to the relative degree of the plant transfer function) are known. In this case, the feedforward compensator is replaced by multiple feedforward actions, one for every requested derivative of the reference input.

For example, if $P(s) = 1/s^2$, then the single feedforward compensator $H_2(s) = s^2$ that performs the double derivative of the reference signal is not practically implementable (it has a non–proper transfer function). However, if the double derivative of the reference signal...
input is known, then it can be directly forwarded, and this avoids the issue of implementing a non–proper feedforward compensator.

- fixed time–delays can be tolerated, provided that the reference input is known in advance over an horizon which is larger than the time delay.

As for the previous case, the feedforward action is implemented by forwarding the the time–advanced replica of the reference signal. For example, if \( P(s) = e^{-sT_d} \), then the feedforward compensation can be performed by directly forwarding the reference input value \( r(t + T_d) \), without requiring to implement any feedforward compensator.

3. compensating known (measured or estimated) input or output disturbances.

With reference to Fig. 1c, it holds that

\[
Y(s) = \frac{C(s)P(s)}{1 + C(s)P(s)} R(s) + \frac{H_3(s)P(s) - P_{yd}(s)}{1 + C(s)P(s)} D(s) \tag{4}
\]

The effect of the known (measured or estimated) output disturbance \( d \) can be cancelled out by choosing

\[
H_3(s) = \frac{P_{yd}(s)}{P(s)} \tag{5}
\]

Similar considerations to those done for (3) regarding the inversion of the plant transfer function are also valid for (5). For the cancellation of a disturbance entering at the plant input, the scheme of Fig. 1d can be adopted, with \( H_4(s) = P_{ud}(s) \).

The feedforward compensations illustrated above can be conveniently applied to improve the performance of the speed PI controller designed in the laboratory activity 2. To this aim, reconsider the simplified model of the DC gearmotor, whose block diagram is reported in Fig. 2a. After some elementary manipulations, the equivalent block diagram of Fig. 2b can be obtained, where the original torque disturbance \( \tau_d \) (static friction torque) is represented as an equivalent input voltage disturbance \( u_d \). With reference to Fig. 2b, it holds that

\[
\Omega_l(s) = P(s) [U(s) - U_d(s)] = P(s) [U(s) - P_{ud}(s) \tau_d(s)] \tag{6}
\]

where

\[
P(s) = \frac{1}{N} \frac{k_m}{T_m s + 1}, \quad P_{ud}(s) = \frac{R_{eq}}{k_{drv} k_t N} \tag{7}
\]

with

\[
k_m = \frac{k_{drv} k_t}{R_{eq} B_{eq} + k_l k_e}, \quad T_m = \frac{R_{eq} J_{eq}}{R_{eq} B_{eq} + k_l k_e} \tag{8}
\]

Given (6), two feedforward compensations can be readily implemented, namely the feedforward compensation of the reference signal of Fig. 1b, and the feedforward cancellation of the input disturbance of Fig. 1d. In Fig. 1b and 1d, \( C(s) \) denotes the speed PI controller, and \( r \) is the speed reference signal \( \omega_l^* \).

Regarding the scheme of Fig. 1b, the feedforward action can be implemented by choosing

\[
H_2(s) = \frac{1}{P(s)} = \frac{N}{k_m (T_m s + 1)} \tag{9}
\]

This feedforward compensator is not implementable in practice, since it requires to compute the derivative of the reference signal. However, when the reference signal is known in advance, the
derivative can be computed offline, and then used to implement the required feedforward action as a simple algebraic expression of the reference signal and its pre–computed derivative. In fact, from (9) it follows that (by inverse Laplace transformation, reminding that $r$ in Fig. 1b is the speed reference signal $\omega^*_l$):

$$u_{ff, 2} = \frac{NT_m}{k_m} \frac{d\omega^*_l}{dt} + \frac{N}{k_m} \omega^*_l$$

(10)

$$= \frac{NR_{eq} J_{eq}}{k_{drv} k_t} \frac{d\omega^*_l}{dt} + \frac{N (R_{eq} B_{eq} + k_t k_e)}{k_{drv} k_t} \omega^*_l$$

(11)

where $d\omega^*_l / dt$ is the precomputed first order derivative of the speed reference $\omega^*_l$, provided that this latter quantity is known in advance.

For the cancellation of the input disturbance $u_d$, assume that $\tau_d$ in Fig. 2b is the static friction torque $\tau_d = \tau_{sf} \text{sign}(\omega_l)$. Then, with reference to the scheme of Fig. 1d, the following feedforward compensation signal can be used

$$u_{ff, 4} = \frac{R_{eq}}{k_{drv} k_t N} \tau_{sf} \text{sign}(\omega^*_l)$$

(12)

By combining (12) with (11), the resulting total feedforward action is

$$u_{ff} = u_{ff, 2} + u_{ff, 4} =$$

$$= \frac{NR_{eq} J_{eq}}{k_{drv} k_t} \frac{d\omega^*_l}{dt} + \frac{N (R_{eq} B_{eq} + k_t k_e)}{k_{drv} k_t} \omega^*_l + \frac{R_{eq}}{k_{drv} k_t N} \tau_{sf} \text{sign}(\omega^*_l)$$

(13)

By rearranging terms, it can be noticed that

$$u_{ff} = \frac{NR_{eq} J_{eq}}{k_{drv} k_t} \frac{d\omega^*_l}{dt} + \frac{R_{eq}}{k_{drv} k_t N} \left[ N^2 B_{eq} \omega^*_l + \tau_{sf} \text{sign}(\omega^*_l) \right] + \frac{N k_e}{k_{drv}} \omega^*_l$$

(14)

In fact, in the friction compensation term, the torque

$$\tau_f = N^2 B_{eq} \omega^*_l + \tau_{sf} \text{sign}(\omega^*_l)$$

(15)
represents an estimate of the total friction torque acting at load side, while the voltage
\[ u_e = N k_e \omega_l^* \] (16)
in the BEMF compensation term is an estimate of the actual BEMF generated by the DC motor at the rotor speed \( N \omega_l^* \) (motor side). On a conventional feedback–based speed PI–control scheme, the feedforward compensation (14) can be introduced as illustrated in Fig. 3.

3 Integrator anti–windup mechanism

In any real control system the actuator output has a limited dynamic range. The block diagram of Fig. 4a takes into account this limitation by including a saturation block between the controller output and the plant input. Whenever the actuator saturates, the command \( u \) provided to the plant is constant, regardless of the tracking error \( e \), so that the feedback loop turns out to be effectively disabled (opened), and both the plant and the PI-controller operate in open–loop.

Suppose that the reference set–point is large enough to cause the actuator to saturate at its upper bound \( \bar{u} = \bar{u}_{\text{max}} \). While the saturation is active, the integrator in the PID controller still continues to integrate the (positive) tracking error \( e \) (integrator windup), and hence the controller output \( \bar{u} \) keeps growing; however, the increasingly growing control command is ineffective for the plant, since it exceeds the actuator limits. When the plant output rises enough to bring the tracking error to zero, the actuator is still saturated because the integrator has (unnecessarily) accumulated a large amount of tracking error in the previous phase. To bring the actuator out of saturation, it is necessary that the tracking error remains negative for enough time to allow the integrator output to come back within the linear operating region of the actuator (integrator unwinding). The effect of the integrator windup on the closed–loop step response is an excessively large overshoot, which is however required to produce the necessary unwinding error for the integrator.

A possible solution to mitigate the integrator windup effects consists of implementing an integrator anti-windup circuit in the PID controller, as shown in Fig. 4b.

A saturation detection mechanism is used within the controller to detect when the actuator is saturated. If the controller is implemented digitally, this mechanism can be implemented as a logical test, such as: “if the PID controller output \( \bar{u} \) exceeds the actuator saturation levels \( \pm \bar{u}_{\text{max}} \), i.e. \( \bar{u} \geq \bar{u}_{\text{max}} \), then the actuator is saturated”. In analog controllers, the easiest way to detect the
Figure 4: PID control in presence of actuator saturation: (a) conventional PID controller; (b) PID controller with anti–windup mechanism.

actuator saturation consists of employing a "simulated saturation" within the controller, and then computing the difference between its input \( \bar{u} \) and output \( u \). If the difference is not zero, a saturation in the controller command is occurring. Obviously, for an effective detection of the actuator saturation, the saturation levels in the 'simulated saturation' block must be matched with those of the real actuator.

When an actuator saturation is detected, the integration has to be either stopped or compensated to avoid the integrator windup issue. In the anti–windup mechanism depicted in Fig. 4b, the integrator input is compensated by a signal proportional to the amount of saturation occurring in the actuator. When a saturation occurs, the local feedback around the integrator becomes active, and the integrator response is modified into that of a first order low–pass filter of the type

\[
H(s) = \frac{1}{s + K_W} \tag{17}
\]

Since the filter is BIBO stable, its output remains bounded (provided that the tracking error \( e \) is bounded), thus preventing the integrator windup phenomenon. In other words, the purpose of the anti–windup mechanism is to provide a local feedback around the integrator in order to make the controller stable alone when the main control loop is inactive (because of the saturation occurring in the actuator). Regarding the choice of the anti–windup gain \( K_W = 1/T_W \), it should be chosen large enough to keep the integrator input small under all error condition. A possible tentative value for the gain is obtained by setting \( T_W = t_{s,5\%}/5 \), where \( t_{s,5\%} \) is the settling time of the control system to a step reference input. However, the initial tentative value may require manual adjustments to get a satisfactory response.
4 Laboratory assignments: numerical simulations

4.1 Integrator anti–windup

1. Reconsider the speed PI controller for the DC gearmotor designed in the laboratory activity 2 (using the motor nominal parameters). On the accurate Simulink model prepared to validate the speed control design, implement the anti–windup (AWU) scheme described in Sec. 3. A possible Simulink implementation of the speed PI controller with AWU is shown in Fig. 5.

The saturation block in Fig. 5b is the “simulated saturation” block used by the controller to detect when a real saturation occurs in the actuator (see also the block diagram of Fig 4b). To determine the saturation levels to set in the “simulated saturation” block, note that from the PI speed controller output $u$ to the motor input voltage (armature voltage) $u_a$ there are basically two physical saturations, namely the DAC and voltage driver saturations. The DAC saturation levels are equal to $\pm 10 \text{ V}$; since the voltage driver DC gain is equal to 0.6, it follows that the driver output never exceed the $\pm 6 \text{ V}$ limits, that are well below the driver output saturation levels of $\pm 12 \text{ V}$. From this simple analysis, it follows that the only saturation that actually limits the control output is that due to the DAC. Therefore, the saturation levels of the “simulated saturation” block will be set equal to those of the DAC, namely $\pm 10 \text{ V}$.

Regarding the anti–windup gain $K_W = 1/T_W$, this should be chosen large enough to keep the integrator input small under all error condition. As a first trial, consider to choose $T_W = t_{s,5\%}/5$, where $t_{s,5\%}$ is the settling time specification used for the design of the PI controller.

2. Validate the effectiveness of the AWU mechanism implemented in point 1, using a step speed reference of amplitude equal to $450 \text{ rpm}$. Compare the overshoot in the step response obtained by enabling or disabling the AWU scheme.

The typical simulation results are reported in Fig. 8.

![Simulink model implementation details](image)

Figure 5: Speed PI controller with anti–windup mechanism: Simulink model implementation details.
4.2 Feedforward compensation

1. In the Simulink model of the speed PI controller with AWU implemented in point 1 of Sec. 4.1, add the feedforward compensation scheme (for friction and inertia compensation) described in Sec. 2. A possible Simulink implementation of the feedforward compensation (14), which corresponds to the block diagram of Fig. 3, is shown in Fig. 6b.

In order to be effective, the feedforward action (14) must be computed by using the values of the friction and inertia parameters (i.e. the values of $\hat{B}_{eq}$, $\hat{\tau}_{sf}$ and $\hat{J}_{eq}$) estimated in the laboratory activity 3.

Please pay attention to use the appropriate measurement units when computing the feedforward compensation. In particular, note that (14) requires that the speed and acceleration references (load side) are specified in $\text{[rad]}$ and $\text{[rad/s]}$ units. Hence, if the reference signals are expressed in different measurement units (typically $\text{[rpm]}$ and $\text{[rpm/s]}$ units), conversions gains must be used before computing the feedforward action. For example, in the implementation of Fig. 6b, two gains are used to convert the speed and acceleration references from $\text{[rpm]}$ and $\text{[rpm/s]}$ units to $\text{[rad]}$ and $\text{[rad/s]}$ units.

Figure 6: Speed PI controller with anti–windup and feedforward compensation: Simulink model implementation details.
Figure 7: Acceleration and speed reference signals used for the feedforward compensation test.

2. Validate the effectiveness of the feedforward compensation scheme implemented in point 1, using a periodic acceleration reference signal with main period defined as follows

\[ a^*(t) = \frac{d\omega^*(t)}{dt} = \begin{cases} 
900 \text{ rpm/s} & \text{if } 0 \leq t < 0.5 \text{ s} \\
0 \text{ rpm/s} & \text{if } 0.5 \leq t < 1 \text{ s} \\
-900 \text{ rpm/s} & \text{if } 1 \leq t < 2 \text{ s} \\
0 \text{ rpm/s} & \text{if } 2 \leq t < 2.5 \text{ s} \\
900 \text{ rpm/s} & \text{if } 2.5 \leq t < 3 \text{ s} 
\end{cases} \]  \hspace{1cm} (18)

which defines a trapezoidal speed profile. The speed reference is obtained by integration of the acceleration profile, i.e.

\[ \omega^*(t) = \int_0^t a^*_i(\tau) d\tau \]  \hspace{1cm} (19)

The resulting acceleration and speed reference signals are shown in Fig. 7. For the simulations, set the parameters \( J_{eq}, B_{eq} \) and \( \tau_{sf} \) in the Simulink model of the DC gearmotor equal to the values \( \hat{J}_{eq}, \hat{B}_{eq} \) and \( \hat{\tau}_{sf} \) estimated in the laboratory activity 3 (and used to compute the feedforward action (14)).

Compare the tracking error obtained by enabling or disabling the feedforward compensation. The typical simulation results are reported in Fig. 9.
Figure 8: Integrator anti–windup mechanism: simulation results

Figure 9: Feedforward compensation: simulation results
5 Laboratory assignments: experimental part

5.1 Integrator anti–windup

1. Validate the anti–windup scheme designed in point 1 of Sec. 4.1 on the experimental setup.

   Perform the experimental tests by adopting the same methodology used for the numerical simulations, namely by following the procedure described in point 2 of Sec. 4.1.

   The typical results of the experimental tests are reported in Fig. 10.

5.2 Feedforward compensation

1. Validate the feedforward compensation scheme designed in point 1 of Sec. 4.2 on the experimental setup.

   Perform the experimental tests by adopting the same methodology used for the numerical simulations, namely by following the procedure described in point 2 of Sec. 4.2.

   The typical results of the experimental tests are reported in Fig. 11.
Figure 10: Integrator anti–windup mechanism: experimental results

Figure 11: Feedforward compensation: experimental results